DM3 : Minimisation d'automates

Ce devoir est facultatif. Il est à rendre pour le 8/11. Si votre objectif est principalement de vous entraîner en C vous pouvez vous restreindre à lire attentivement et comprendre les algorithmes de la partie 2, en traiter les questions 12 et 16, puis traiter la partie 3.

Partie 1 *Automate des résiduels*

Dans cette partie, Σ désigne un alphabet non vide. Si u est un mot de $\Sigma^\star$ et L un langage sur Σ, on appelle *résiduel de L par rapport à u* le langage noté $u^{-1}L$ tel que

$$u^{-1}L = \{m \in \Sigma^\star \mid um \in L\}$$

Intuitivement le résiduel de L par rapport à u est l'ensemble des mots avec lesquels on peut compléter u pour obtenir un mot de L. L'ensemble $\mathcal{R}_L = \{u^{-1}L \mid u \in \Sigma^\star\}$ s'appelle *l'ensemble des résiduels de L*.

1. Si L est un langage, quel est le résiduel de L par rapport à ε ?
2. Montrer que pour tout langage L sur Σ et tous mots $u, v \in \Sigma^\star$, $u^{-1}(v^{-1}L) = (vu)^{-1}L$.
3. a) Considérons le langage L_{ab} dénoté par $(a+b)^\star ab(a+b)^\star$. Déterminer le résiduel de L_{ab} par rapport à bab puis par rapport à $aaaba$. Que constate-t-on ?

   b) Expliquer pourquoi l'ensemble des résiduels de L_{ab} est $\{(a+b)^\star, L + b(a+b)^\star, L\}$.

Un premier objectif de cette partie est de montrer le résultat suivant : un langage L est reconnaissable si et seulement si il admet un nombre fini de résiduels. Remarquons que cette preuve fournira une nouvelle caractérisation des langages rationnels : ce sont ceux qui ont un nombre fini de résiduels.

4. Soit L un langage reconnaissable. Alors il l'est par un automate $A = (\Sigma, Q, q_0, F, \delta)$ qu'on peut supposer déterministe, complet et dont tous les états sont accessibles sans perte de généralité.

   a) Montrer que la fonction $\varphi : \begin{cases} Q \longrightarrow \mathcal{R}_L \\ q \longmapsto u^{-1}L \text{ où } u \text{ est l'un des mots tels que } \delta^\star(q_0, u) = q \end{cases}$

   est correctement définie (il faudra en particulier montrer que $\varphi(q)$ ne dépend pas du choix du mot u tel que $\delta^\star(q_0, u) = q$).

   b) A l'aide de la fonction φ, montrer que L admet un nombre fini de résiduels.

5. Soit à présent L un langage ayant un nombre fini de résiduels. On définit alors un automate $M(L) = (\Sigma, Q, I, F, \delta)$, appelé *automate des résiduels associé à L*, comme suit :
   - Q est l'ensemble des résiduels de L.
   - I contient un seul élément : $\varepsilon^{-1}L$.
   - $F = \{u^{-1}L \mid u \in L\}$.
   - pour toute lettre $a \in \Sigma$ et tout résiduel $u^{-1}L \in Q$, $\delta(u^{-1}L, a) = (ua)^{-1}L$.

   a) Montrer que si $u^{-1}L = v^{-1}L$, alors pour tout $a \in \Sigma$, $\delta(u^{-1}L, a) = \delta(v^{-1}L, a)$. Cette propriété garantit la bonne définition de la fonction de transition δ.

   b) Montrer que $M(L)$ est un automate fini déterministe et complet.

   c) Montrer que $M(L)$ reconnaît le langage L.

6. A l'aide des questions précédentes, déterminer l'automate des résiduels associé au langage L_{ab}.

Un automate reconnaissant un langage L est dit *minimal* s'il est déterministe, complet et possède le plus petit nombre d'états possible parmi les automates déterministes et complets reconnaissant L.

7. Montrer que l'automate des résiduels est un automate minimal.

Cette dernière question montre en particulier l'existence d'un automate minimal pour n'importe quel langage rationnel. On peut montrer que cet automate minimal est en fait unique (à renommage des états près) ce qui fournit une façon de déterminer si deux automates reconnaissent le même langage : il suffit de les minimiser et comparer les deux automates obtenus. Reste une question : comment déterminer algorithmiquement l'automate minimal équivalent à un automate donné ?

Partie 2 *Congruence de Nerode*

Dans cette partie, $A = (\Sigma, Q, q_0, F, \delta)$ est un automate complet et déterministe reconnaissant un certain langage L. Une *congruence sur* A est une relation d'équivalence $\sim$ sur Q vérifiant de surcroît les deux propriétés suivantes :

(1) Pour tous $q, q' \in Q$, si $q \sim q'$ alors ($q \in F$ si et seulement si $q' \in F$).

(2) Pour tous $q, q' \in Q$, si $q \sim q'$ alors pour tout $a \in \Sigma$, $\delta(q, a) \sim \delta(q', a)$.

Si $\sim$ est une congruence sur A, on note $\overline{q}$ la classe de l'état q pour cette relation d'équivalence. *L'automate quotient selon* $\sim$, noté $A/\sim$, est l'automate défini par $(\Sigma, Q/\sim, \overline{q_0}, \overline{F}, \overline{\delta})$ avec $\overline{F} = \{\overline{q} \,|\, q \in F\}$ et pour tout $\overline{q} \in Q/\sim$ et tout $a \in \Sigma$, $\overline{\delta}(\overline{q}, a) = \overline{\delta(q, a)}$.

8. On suppose que $\sim$ est une congruence sur A.

   a) Montrer que $A/\sim$ est un automate correctement défini.

   b) Montrer que $A/\sim$ reconnaît le même langage que A.

On s'intéresse à présent à une congruence particulière : la congruence de Nerode. Dans le suite, le symbole $\sim$ désignera toujours cette congruence. Elle est définie comme suit :

$$q \sim q' \text{ si et seulement si } L_q = L_{q'}$$

où pour tout $q \in Q$, $L_q = \{u \in \Sigma^\star \,|\, \delta^\star(q, u) \in F\}$. Autrement dit, deux états sont équivalents selon la congruence de Nerode si les mots reconnus à partir de q sont les mêmes que ceux reconnus à partir de q'.

9. Montrer que la congruence de Nerode... est effectivement une congruence.

10. Montrer que pour tout automate A déterministe, complet et accessible, l'automate quotient $A/\sim$ est l'automate minimal permettant de reconnaître le langage $L(A)$.

Cette partie montre que pour minimiser un automate, il suffit de savoir déterminer les classes d'équivalence pour la congruence de Nerode. On va calculer les classes d'équivalence d'états selon cette relation par raffinements successifs. On introduit pour ce faire, pour tout $k \in \mathbb{N}$, la relation d'équivalence $\sim_k$ sur Q définie comme suit :

$$q \sim_k q' \text{ si et seulement si } \{u \in L_q \,|\, |u| \leq k\} = \{u \in L_{q'} \,|\, |u| \leq k\}$$

11. Déterminer les classes d'équivalence d'états de A selon $\sim_0$.

12. Montrer que pour tous $q, q' \in Q$ et tout $k \in \mathbb{N}$: $q \sim_{k+1} q' \Leftrightarrow (q \sim_k q' \text{ et } \forall a \in \Sigma, \delta(q, a) \sim_k \delta(q', a))$.

13. Montrer que si $\sim_{k+1}$ et $\sim_k$ coïncident (c'est-à-dire, fournissent les mêmes classes d'équivalence d'états), alors pour tout $l > 0$, $\sim_{k+l}$ et $\sim_k$ coïcident.

14. En déduire que, si $\sim_k$ et $\sim_{k+1}$ sont égales alors $\sim_k$ et $\sim$ sont égales.

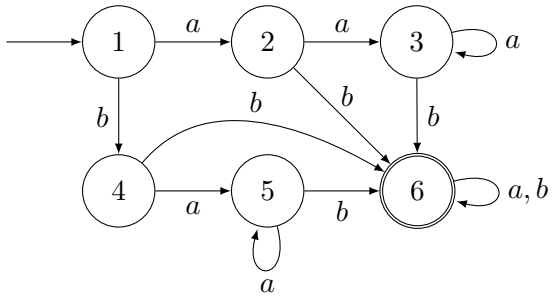
Le processus décrit aux questions 12 à 15 permet ainsi de calculer la congruence de Nerode de proche en proche, à condition qu'on ait effectivement une stabilisation des relations d'équivalence $\sim_k$.

15. Expliquer pourquoi $\sim_{|Q|-2}$ est égale à $\sim$.

Cette dernière question garantit la terminaison du processus suivant :

Entrée : Un automate A déterministe et complet.
Sortie : Les classes d'équivalence d'états de A selon $\sim$.
nerode(A) =
    Déterminer les classes selon $\sim_0$ via la question 12
    En déduire les classes selon $\sim_1$ via la question 13
    $i \leftarrow 0$
    Tant que $\sim_i$ et $\sim_{i+1}$ ne coïncident pas
        Calculer $\sim_{i+2}$ à partir de $\sim_{i+1}$ et de la question 13.
        $i \leftarrow i + 1$
    Renvoyer les classes selon $\sim_i$.

16. Minimiser l'automate A ci-dessous. On commencera par calculer en détail les classes d'équivalence de Nerode à l'aide de l'algorithme précédent. Quel langage A reconnaît-il ?



Partie 3 *Minimisation via l'algorithme de Moore*

Dans cette partie, on cherche à implémenter un algorithme de minimisation d'automate en C. On utilise pour ce faire l'algorithme de Moore dont le principe est le suivant :

- Calculer successivement les relations $\sim_0, \sim_1$ ... sur les états de A jusqu'à stabilisation avec l'algorithme nerode décrit en fin de partie 2. On obtient ainsi la relation $\sim$.
- Calculer l'automate l'automate quotient $A/\sim$ défini en partie 2.

Dans toute cette partie les automates considérés sont déterministes, complets et accessibles et on les représente comme dans le TP5 (avec les mêmes conventions) à l'aide de la structure suivante :

```
struct AFD {
    int taille_Q;
    int taille_Sigma;
    int q0;
    bool* finaux;
    int** delta;
};
typedef struct AFD afd;
```

On pourra s'aider des fonctions implémentées dans la partie 1 du TP5 pour construire (et libérer correctement !) des automates sur lesquels faire des tests des fonctions à implémenter dans ce devoir mais il est interdit de les utiliser dans le corps desdites fonctions.

Pour représenter les classes d'équivalence d'une relation $\sim_k$ sur un automate à n états, on utilisera un tableau d'entiers de taille n dont la case i contient le numéro de la classe dans lequel est l'état i. On impose les deux contraintes ($\star$) suivantes sur un tel tableau :

- Les classes sont numérotées de 0 à $c-1$ où c et le nombre de classes selon $\sim_k$.
- Les numéros des classes sont attribuées par ordre croissant de numéro d'état. Plus précisément : l'état 0 sera dans la classe numéro 0. Le plus petit état qui n'est pas dans la classe 0 sera dans la classe 1. Le plus petit état qui n'est ni dans la classe 0 ni dans la classe 1 sera dans la classe 2...

17. On considère un automate à 8 états et une relation d'équivalence sur ces états dont les classes sont $\{2,5,6\}$, $\{0,1,3,7\}$ et $\{4\}$. Dans chaque cas, dire (en justifiant si ce n'est pas le cas) si le tableau proposé représente ces classes d'équivalence.

    a) $[0,0,1,0,2,1,1,0]$.

    b) $[0,0,2,0,1,2,2,0]$.

    c) $[0,0,1,2,2,1,1,0]$.

Les questions suivantes visent à implémenter une version naïve de l'algorithme de Moore. On n'hésitera pas à introduire des fonctions auxiliaires dont on expliquera le fonctionnement et l'intérêt pour clarifier la construction des fonctions exigées.

18. Ecrire une fonction `int* initialiser_classes(afd *a)` prenant en entrée un automate A et renvoyant un tableau alloué sur le tas représentant les classes d'équivalence d'états de A selon la relation $\sim_0$.

19. Ecrire une fonction `int classe_d_arrivee(afd* a, int* c, int q, int l)` qui indique quelle est (le numéro de) la classe de l'état dans lequel on arrive en lisant la lettre `l` depuis l'état `q` dans l'automate `a` lorsque les classes d'équivalence sont données par `c`.

20. Ecrire une fonction `bool distinguables(afd* a, int* c, int q, int r)` prenant en entrée un automate `a` et le tableau `c` représentant les classes d'équivalence selon la relation $\sim_k$ et renvoyant `false` si les états `q` et `r` de `a` sont dans la même classe pour la relation $\sim_{k+1}$ et `true` sinon.

21. Ecrire une fonction `int* raffiner(afd* a, int* c)` prenant en entrée un automate `a` et le tableau `c` représentant les classes d'équivalence d'états de cet automate selon $\sim_k$ et renvoyant un tableau qui encode les classes selon $\sim_{k+1}$. On justifiera que le tableau construit vérifie les contraintes $(\star)$.

22. Ecrire une fonction `int* nerode(afd* a)` renvoyant le tableau représentant les classes d'équivalence d'états de `a` selon la congruence de Nerode. On expliquera la démarche.

23. Ecrire enfin une fonction `afd* minimiser(afd* a)` construisant l'automate minimal associé à un automate déterministe, complet et accessible donné en entrée selon l'algorithme de Moore.

24. Déterminer la complexité de `minimiser` en fonction de la taille n de l'automate en entrée et du nombre de lettres p de son alphabet. On détaillera le raisonnement.

Bonus *Procédé de minimisation de Brzozowski*

Si A est un automate fini, le transposé de A, noté $T(A)$ est l'automate dans lequel tous les arcs de A sont inversés et dans lequel les états initiaux deviennent finaux et les finaux initiaux. Le déterminisé de A, noté $D(A)$ est l'automate obtenu à partir de A par déterminisation accessible.

25. Montrer que, si A est un automate qui reconnaît L tel que $T(A)$ est déterministe et accessible alors $D(A)$ est l'automate minimal reconnaissant L. *Indication : s'inspirer des idées de la question 4.*

26. En déduire la correction du processus de minimisation de Brzozowski :

> **Entrée :** Un automate A reconnaissant L.
> **Sortie :** L'automate minimal reconnaissant L.
> minimisation_Brzozowki(A) =
> Renvoyer $D(T(D(T(A))))$

27. Appliquer cet algorithme à l'automate A de la question 17 et vérifier que l'automate obtenu est bien égal à $A/\sim$ (à renommage près des états).